

```

5      /**<SCRIPT LANGUAGE="JavaScript1.2"> */

10     //////////////////////////////////////
    // Source Safe version and Date: DO NOT REMOVE
    // $Header: /Versions/Current/Source/Collection/Client/ePulse/ePulse.js 45      4/10/00 10:34a
    Rob $
    //
15    // DO NOT REMOVE the LOGFILE and REVISION below. They are redundant
    // with the $Header, but they are necessary because the Obfuscator will
    // pass them through to the obfuscated script.
    // $Logfile: /Versions/Current/Source/Collection/Client/ePulse/ePulse.js $
    // $Revision: 45 $
    //////////////////////////////////////
20
    var kl_isIE = navigator.appName.indexOf("Microsoft")!=-1;
    var kl_isNav = navigator.appName.indexOf("Netscape")!=-1 &&
        parseInt(navigator.appVersion) >= 4;
25    if (kl_isIE || kl_isNav)
    {
        /** BEGIN Customer Specific Information */
        var kl_rootpath = "__rootpath__";
        var kl_companyid = "__companyid__";
        var kl_siteid = "KL_siteid__";
30        var kl_collectiongetUrl = "__collectiongetUrl__";

        /** These parameters are used in the following way:
        kl_rootpath is used to set the visibility of the session cookie.
35        Its value is normally "/". But if there are two sites within the same
        domain, it may be different. For example, say there are two logical
        sites, one at http://ebusiness/Lucky and one at http://ebusiness/StaterBros,
        then to track sessions on these sites separately, the Lucky site would
        have to have its rootpath set to /Lucky and the StaterBros site would
40        have to have its rootpath set to /StateBros.
        kl_siteid is the site Id in the database with the string KL prepended
        kl_collectiongetUrl is the image file that the script does a "GET" on in
        order to send data to the server.
        */
45        /** END Customer Specific Information */

        // this is a new one that should be added
        var kl_sessionCookieVisibility = ""; // this can be "" in most cases
        var kl_sessionCookiePathVisibility = "; path=" + kl_rootpath;
50        var kl_pageViewKey = "KL_PV"+kl_siteid;

        var kl_sessionCookieKey = kl_siteid;
        // Record in the window the kl_startload time for the entire page, it
        // it isn't already set.
55        // And record in this window the start load time of itself
        var kl_startload = new Date();
    }

60    // Retrieve the named value from the cookie. If the
    // value doesn't exist, return ""
    function kl_getCookieValue(kl_valueName, kl_cookie)
    {
65        // s is start index
        var s = kl_cookie.indexOf(kl_valueName);
        if (s == -1)
            return "";
        s += kl_valueName.length;
70        // e is end index
        var e = kl_cookie.indexOf(";", s);

```



```

5         if ((kl_isNav || kl_ofuncBody.indexOf("anonymous") > 0) && b >
0)
            kl_ofuncBody = kl_ofuncBody.substring(b,
kl_ofuncBody.length);
            else if (kl_ofuncBody.indexOf("function") >= 0 && b > 0)
10         kl_ofuncBody =
kl_ofuncBody.substring(kl_ofuncBody.indexOf("function") + 9, b);

        // escape any newline chars in the old function body, to exscaped
15 newlines
        kl_ofuncBody = kl_ofuncBody.replace(new RegExp("\\n","g"), "\\n");
        // replace any occurences of the this pointer to kl_this. The newly
defined
        // user function will be passed the original this pointer as
20 kl_this. This is
        // necessary since the scope of the original code is being rplaced.
        kl_ofuncBody = kl_ofuncBody.replace(
            new RegExp("([a-zA-Z0-9$_])this([a-zA-Z0-9$_])","g"),
"$1kl_this$2");

        //kl_ofuncBody = kl_ofuncBody.replace(/\\n/g, " ; ");
25 // Escape all ' characters
        kl_ofuncBody = kl_ofuncBody.replace(new RegExp("'", "g"), "\\'");
        // escape all " characters
        kl_ofuncBody = kl_ofuncBody.replace(new RegExp('"', "g"), "\\\"");

30 // create then function body for the new event handler.
        var kl_newFuncBody =
            "{ " +
            // "window.alert('START ');" +
            "var a0 = '" + a + "';" +
            "var ofb = '" + kl_ofuncBody + "';" +
            "var f = new Function( a0, 'kl_this', ofb);" +
            "var rv = f( " + a + ",this);" +
            kl_method + " ;" +
            // "window.alert('FINISHED !!!!!!!!!!!!!!!1');" +
            "return rv;" +
            " }";

40
        return new Function(a, kl_newFuncBody);
45     }
        else
            return kl_oldFunc;
    }
50     return kl_newFunc;
}

// force conversion to string
55 function kl_rts(URL)
{
    // URL MUST be either of type String or Location
    return URL.toString();
60 }

function kl_logerror(rs,hf,ln)
{
65     // m stands for message throughout
    var m = "";

    if (rs.target != null)
    {
70         m += "&rs=" + escape("Image load error.");
        m += "&hf=" + escape(kl_rts(rs.target.src.split("?")[0]));
    }
    else

```

```

5      {
        m += "&rs="+escape(rs);
        if (hf != document.location)
            m += "&hf="+escape(kl_rts(hf));
10     if (ln > 0)
            m += "&ln="+ln;
        }
        kl_log("ERROR", m);

15     return true;
    }

20 function kl_startmonitor(e)
    {
        // w mean win
        var w = (kl_isIE ? window : e.target);

25     // if the window has an instrumentation function, instrument it.
        // this will bypass calling instrument for frameset windows, but
        // all others will be instrumented.
        //if (win.kl_instrument != null)

30     w.kl_instrument();

        // If there's data to send, send it. Normally there is data only if this is the first page
        // loaded in the session, and the SYSTEM record is what gets sent here.
        if (kl_amFirstWindowLoaded && !(kl_sessionrec == null))
35     {
            kl_submitlog("echo=" + kl_sessionrec);
            kl_sessionrec = null;
        }

40     kl_log("LOAD", "");
    }

45 function kl_addfocushandlers(o)
    {
        o.onfocus = kl_setHandle("kl_startfocus(e)", kl_startfocus, o.onfocus);
        o.onblur = kl_setHandle("kl_endfocus(e)", kl_endfocus, o.onblur);
50    }

    function kl_startfocus(e)
    {
        var el = (kl_isIE ? window.event.srcElement : e.target);

55     el.kl_focusstart = new Date();
        if (el.form != null)
        {
            if (el.name)
                el.form.myPeer.le = el.name;
60         else if (el.id)
            el.form.myPeer.le = el.id;
        }

65    }

    function kl_endfocus(e)
    {
        var el = (kl_isIE ? window.event.srcElement : e.target);
        var kl_interval = 0;
70     var now = new Date();

        if (el.kl_focusstart != null)
        {

```

002090"96278560  
0587236-060200

```
5      kl_interval += now.getTime() - el.kl_focusstart.getTime();
      el.form.myPeer.ft += kl_interval;
      if (el.kl_isSet == false)
      {
10         el.form.myPeer.ns += 1;
         el.kl_isSet = true;
      }

      el.kl_myElementPeer.kl_value = kl_extractValue(el);

15  }
    return true;
  }
}

20 function kl_gotolink(e)
{
    // l is link
    var l = (kl_isIE ? window.event.srcElement : e.target );
    var isad = false;

25    if (kl_isIE)
    {
        var uc = l.tagName.toUpperCase();
        isad = (uc == "IMG" ? true : false );

30        while (uc.indexOf("A"))
        {
            l = l.parentElement;
            uc = l.tagName.toUpperCase();

35        }
    }

    var msg = "&lk="+escape(kl_rts(l));
    if (isad)
40        msg += "&ad=YES";

    kl_log("GOTO", msg);

45 }

function kl_submitform(e)
{
    // f is form
    var f = (kl_isIE ? window.event.srcElement : e.target);

50    f.kl_lastelement = 0;
    f.kl_logformimpression("SUBMIT", f.myPeer);

55 }

function kl_resetform(e)
{
60    // f is form
    var f = (kl_isIE ? window.event.srcElement : e.target);

    f.kl_lastelement = 0;
    f.kl_logformimpression("RESET", f.myPeer);

65 }

function kl_logformimpression(reason, peer)
70 {
    // NOTICE the with statement. The peer attributes used within are:
    // ne, name, ns, le ft
    with (peer)
    {
```

002030-98278560  
09587236-060200

```
5      if (ne > 0)
      {
10         var m = "&rs=" + reason; // m is message

        // if the form has a name, report on it.

        m += "&nm=" + escape(name);

        // if the focus time of the form was > 0, then report on it.
15         if ( ft > 0 )
        {
            m += "&ne=" + ne;
            if (ns > 0)
                m += "&ns=" + ns;
            if (le != -1)
20                m += "&le=" + le;

            m += "&ft=" + ft;

            // add the user-defd fields if there are any
            // the number of fields is limited to 10.
            var lim = 0;
            if (kl_InfoAry != null)
                lim = kl_InfoAry.length;

30            if (lim > 10)
                lim = 10;
            for (var i = 0; i < lim; i++)
                if (kl_InfoAry[i] != null)
                    m += "&i" + i.toString() + "=" + kl_InfoAry[i].toString();

35            kl_log("FORM",m);
            // reset the peer fields
            ft = 0;
            ns = 0;
            le = -1;
40        }
    }
}

45 function kl_abortload(e)
{
    var img = (kl_isIE ? evendow.event.srcElement : e.target);

50    var m = "&ig=" + escape(img.src);

    kl_log("ABORT", m);

55 }

function kl_checkabort()
{
    this.imgabort = 0;
    if (kl_isIE && document.images)
60    {
        var ng = 5;
        for (var i = 0; i < document.images.length; i++)
        {
            var img = document.images[i];
            if (img.complete == false )
65            {
                this.imgabort++;
                if (ng-- > 0)
                    kl_log("ABORT", "&ig="+escape(img.src));
70            }
        }
    }
}
```

```

5 function kl_download(e)
{
    if (kl_formAry != null)
    {
10         for (var d = 0; d < kl_formAry.length; d++)
            kl_logformimpression("ABANDON["+d+"]", kl_formAry[d]);
    }

    if (kl_checkabort != null)
        kl_checkabort();

    kl_log("UNLOAD", (imgabort > 0 ? "&ab="+imgabort : ""));
}

20 function kl_dobeforeunload(e)
{
    var win = (kl_isIE ? window : e.target);

    // if there's nothing to report just let it pass
    if (win.kl_logvalue == null)
        return;

    kl_download();

    if ( !(window.kl_logvalue == null) )
        kl_submitlog("echo=" + window.kl_logvalue);

    if (kl_isIE)
        window.onbeforeunload = null;
}

35 function kl_timestamp(stamp, sys)
{
    var m = "&b=TSTAMP&st=" + escape(stamp.toGMTString())
        + "&zn=" + (stamp.getTimezoneOffset() * -1) // need to reverse the sign
        + "&cd=" + kl_companyid + "&sd=" + kl_siteid + "&id=" + kl_theTopWin.klid + "&e=";

    var doc = document;

    if (sys && kl_inframe)
        doc = kl_theTopDoc;

    m+="&b=PAGE";
    if (doc.location)
        m += "&lc="+escape(kl_rts(doc.location));

55 if (!sys && kl_inframe)
    {
        m += "&pt="+escape(kl_rts(kl_theTopWin.location));
    }

    if (doc.location.href != doc.location)
        m += "&hf="+escape(kl_rts(doc.location.href));
    if (doc.referrer)
        m += "&rf="+escape(kl_rts(doc.referrer));
    m += "&sq=" + escape(kl_theTopWin.klsq.toString());
    //m += "&sq=" + escape("1");

65 m += "&e=";

    // dink the pageview sequence number in the cookie

    kl_theTopWin.klsq += 1;
    // rewrite the cookie
    kl_theTopWin.document.cookie =

```





```

5      // set up to form data locations on the window. These locations will be used to store
      // collected information about the forms. This is necessary, only because under Netscape,
      // by the time the window receives an unload event the document is (often) destroyed.

10     // hold the array of form peers
      w.kl_formArray = new Array();

      // holds slots 0 thru 9 of user-specified data (usually form data) for the page
15     w.kl_InfoArray = new Array();

      // set up the forms handlers on all interior forms
      for (var j = 0; j < d.forms.length; j++)
      {
20         var m = d.forms[j]; // just to save on characters

        // A form doesn't get focus!
        //kl_addfocushandlers(m);

        // set up the form info array on the form

25         // set up the window storage area
        w.kl_formArray[j] = new Object();
        var n = w.kl_formArray[j]; // just to save on characters

30         // make the form aware of the storage area that maintains his storage
        m.myPeer = n;

        // ALERT, ALERT, ALERT
        // if a form has an element named "name" (a very stupid thing to do, but it
35         // has been done) then the element overrides the
        // form.name property and makes the form name completely inaccessible to us.
        // Thus before recording the form name we have to discover
        // if its a string object and its not null

40         if (m.name != null && typeof m.name == "string")
            n.name = m.name;
        else
            n.name = j;

45         n.ne = m.elements.length;
        n.ns = 0;
        n.le = -1;
        n.ft = 0;
        n.elems = new Array();

50         m.onsubmit = kl_setHandle("kl_submitform(e)", kl_submitform, m.onsubmit);
        m.onreset = kl_setHandle("kl_resetform(e)", kl_resetform, m.onreset);

        for (var e = 0; e < m.elements.length; e++)
        {
55             kl_addfocushandlers(m.elements[e]);
            // set up the storage area
            n.elems[e] = new Object();
            // setup the element peer
            m.elements[e].kl_myElementPeer = n.elems[e];
            //n.elems[e].accumfocus = 0;
            n.elems[e].kl_focusstart = -1;
            n.elems[e].kl_isSet = false;
65             n.elems[e].kl_value = kl_extractValue(m.elements[e]);
            n.elems[e].kl_name = m.elements[e].name;
        }
        m.kl_logformimpression = kl_logformimpression;

70     }
}

function kl_extractValue(e) // e is an input element on a form
{

```

002090" 9E.248560  
05587236 . 060200

```
5      // extract the "current value" of the input element. Different "types" of input elements
      // are treated differently.

      // weed out bad params -- this code might not be necessary
10     if (e == null || e.type == null)
        return null;

        var t = e.type.toLowerCase();
        if (t == "checkbox")
15         {
            return e.checked;
        }
        else if (t == "text")
        {
20         return e.value;
        }

        // recording values from other input types (such as SELECT are not supported at the moment)

25     return null;
    }

function kl_startsession()
30 {
    var w = window;
    var s = w.screen, n = navigator;
    var msg = "&sc="
        +s.width+"x"+s.height+"x"+s.colorDepth
35     + "&bw="+escape(n.appName)+"&vs="+escape(n.appVersion)
        + "&pf="+escape(n.platform)+"&ua="+escape(n.userAgent);
    kl_log("SYSTEM", msg);
    w.kl_sessionrec = w.kl_logvalue;
    w.kl_logvalue = null;
40 }

/** START ePulse Initialization **/

45 // IMPORTANT the ';' below is NECESSARY for Netscape to be compatible with the
// obfuscated script
// DON NOT REMOVE//////////////////////////////////////
50 ;
//////////////////////////////////////

if (kl_isIE || kl_isNav) // only allow Nets
55 {
    var kl_theTopWin = window;
    var kl_theTopDoc = document;

60     // kl_inframe == true means we are not the top window
    var kl_inframe = (top != window);

    // define several property on this window. This is just to make sure
    // we don't have to deal with "undefined" value
65     var kl_logvalue = null;
    var kl_sessionrec = null;
    var kl_useapplet = false;

70     var kl_amFirstWindowLoaded = false;
    if (kl_isNav)
    {
        kl_useapplet = true;
        // use the base part of the collection URL as the code base
    }
}
```

```

5      var cb = kl_collectiongetURL.substr(0, kl_collectiongetURL.lastIndexOf("/"));

      document.writeln("<APPLET code=ePingA.class codeBase=\"\" + cb + \"\" id=ePingA\"+
        \" name=ePingA width=1 height=1 align=right MAYSCRIPT></APPLET>");
10    }
    else
        kl_useapplet = false;

    // IMPORTANT: For netscape, this line HAS to come after the call to document.writeln(),
    above.
15    // It seems that document.writeln() causes netscape to set up some document properties that
    // otherwise wouldn't have been accessible until after the onload event occurs.
    // If you take the call to document.writeln(), above, out the following statment will cause
    // some kind of unhandable (in javascript) error such as access is denied.
    var kl_isSessionStarted = kl_getCookieValue(kl_siteid+"=", document.cookie);

20    // kl_isSessionStarted is used to determine if this is the first (or one of the first in
    framesets)
    // page view in the session. if there is not yet a session cookie, then this is one of the
    first (
25    // (or cookies might be off). If this is the first, then walk the hierarchy trying to get
    at
    // the highest accessible (in same domain) parent window.

    //var ses = kl_getCookieValue(kl_siteid+"=", document.cookie);
30    if ( kl_isSessionStarted == "" ) // only try to find the real top win if there is no
    session cookie
    {
        // keep going only until we reach the top or our referrer is in a different domain.
        while(kl_theTopWin != top && kl_inDomain(kl_theTopDoc.referrer))
35        {
            kl_theTopDoc = kl_theTopWin.parent.document;
            kl_theTopWin = kl_theTopWin.parent;
        }
    }

40    // this processing relevant for fames. One window (the first window loaded)
    // is chosen to set up and determine the session housekeeping on the top window.
    //if (top.seenFirst == null)
    if ( typeof kl_theTopWin.seenFirst == "undefined" || kl_theTopWin.seenFirst == null)
45    {
        kl_theTopWin.seenFirst = true;
        kl_amFirstWindowLoaded = true;
    }

50    if (kl_amFirstWindowLoaded)
    {

        // The first window loaded sets up some things on the top window
        kl_theTopWin.kl_companyid = this.kl_companyid;
        kl_theTopWin.kl_siteid = this.kl_siteid;
        // set up the session id and sequence number storage areas.
        kl_theTopWin.klid = "";
        kl_theTopWin.klsq = "";
60        if (kl_theTopWin.document.cookie != null)
        {
            kl_theTopWin.klid = kl_getCookieValue(kl_siteid+"=", kl_theTopWin.document.cookie);
            kl_theTopWin.klsq =
                        parseInt(kl_getCookieValue(kl_pageViewKey+"=",
70            kl_theTopWin.document.cookie));
        }

        if (kl_theTopWin.klid == "")
        {
            //window.alert("Should be starting session");
            // take care of the session cookie
            kl_theTopWin.klsq = 0;
            kl_theTopWin.klid = "x" + kl_startload.getTime().toString();

```

002090" 96249560

```
5      kl_theTopWin.klid = kl_theTopWin.klid + Math.random().toString().substr(2,20-
kl_theTopWin.klid.length);
      kl_theTopWin.document.cookie= kl_siteid+"="+kl_theTopWin.klid +
kl_sessionCookiePathVisibility +
10      kl_sessionCookieVisibility;
      kl_theTopWin.document.cookie= kl_pageViewKey+"=0" +kl_sessionCookiePathVisibility +
      kl_sessionCookieVisibility;
      kl_startsession();
15  }
  }

  // the "this" qualification below is necessary. Some browsers
  // will require it, don't remove.
20  this.onabort = kl_setHandle("kl_abortload(e)", kl_abortload, this.onabort);
  this.onerror = kl_setHandle("kl_logerror(rs,hf,ln)", kl_logerror, this.onerror);
  this.onload = kl_setHandle("kl_startmonitor(e)", kl_startmonitor, this.onload);
  if (kl_isNav)
25    window.captureEvents(Event.ABORT | Event.ERROR );

  if (kl_isIE)
    // if not in a frame and IE, use the onbeforeunload handler
    onbeforeunload = kl_setHandle("kl_dobeforeunload(e)", kl_dobeforeunload,
30    onbeforeunload);
  else // netscape doesn't have on beforeunload, so we are stuck with onunload (but the
  applet
    // delivery mechanism solves the problems this presents.
    this.onunload = kl_setHandle("kl_dobeforeunload(e)", kl_dobeforeunload, this.onunload);
35
    // If the controller window (the amTopdog) is in a frame,
    // we have to adjust the event handling because the top dog might get
    // destroyed if his frame content changes.
  }
40

  /** END ePulse Initialization */
  ////////////////////////////////////////
  // WARNING: Do not change this copyright notice at all. It is a tag string
  // that is looked for by the iEchoes Web server module Farm Monitoring process
45  /* Copyright 1999, 2000 Keylime Software, Inc. All Rights Reserved */
  ////////////////////////////////////////

  /**<SCRIPT> */
50
```

226289 v1/SD  
6701!.DOC  
060200/1644